PharmaSuite®

DCS ADAPTER

RELEASE 11.00.00

TECHNICAL MANUAL

PUBLICATION DCTMAD-GR004A-EN-E-DECEMBER-2022
10006963777/PUB
Supersedes publication DCTMAD-GR003A-EN-E

Rockwell Automation

Contact Rockwell See contact information provided in your maintenance contract.

Warranty This product is warranted in accordance with the product license. The product's performance may be affected by system configuration, the application being performed, operator control, maintenance, and other related factors. Rockwell Automation is not responsible for these intervening factors. The instructions in this document do not cover all the details or variations in the equipment, procedure, or process described, nor do they provide directions for meeting every possible contingency during installation, operation, or maintenance. This product's implementation may vary among users.

This document is current as of the time of release of the product; however, the accompanying software may have changed since the release. Rockwell Automation, Inc. reserves the right to change any information contained in this document or the software at any time without prior notice. It is your responsibility to obtain the most current information available from Rockwell when installing or using this product.

Industry Terminology Rockwell Automation recognizes that some of the terms that are currently used in our industry and our publications are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Contents

Figures

Chapter 1

Introduction

This manual is intended for system integrators who connect a Manufacturing Execution System (MES) with one or more Distributed Controls Systems (DCS) by means of the DCS Adapter 2.5.0.

The DCS Adapter provides the possibility to send standardized requests between the MES and any DCS and to receive standardized replies.

For ease of handling, typically a middleware component (Manufacturing Service Bus (MSB)/Enterprise Service Bus (ESB)) is used to transform the MES requests into the data structure and technology of a specific DCS.

This chapter details the supported integration touchpoints (page 3); the following chapters describe the architecture of an MES integrated with the DCS Adapter (page 5), the integration into an MES (page 9), the integration of a DCS (page 23), and how to extend the standard (page 55).

Typographical Conventions

This documentation uses typographical conventions to enhance the readability of the information it presents. The following kinds of formatting indicate specific information:

Bold typeface Designates user interface texts, such as

- window and dialog titles
- menu functions
- panel, tab, and button names
- box labels
- object properties and their values (e.g., status).

Italic typeface Designates technical background information, such as

- path, folder, and file names
- methods
- classes.

CAPITALS	Designate keyboard-related information, such as ■ key names ■ keyboard shortcuts.
`Monospaced typeface`	Designates code examples.

List of Abbreviations

In this document, the following abbreviations are used:

Abbreviation	Definition
B2MML	Business to Manufacturing Markup Language
DCOM	Distributed Component Object Model
DCS	Distributed Control System
ESB	Enterprise Service Bus
JAR	Java Archive
JMS	Java Message Service
MES	Manufacturing Execution System
MSB	Manufacturing Service Bus
S88	ANSI/ISA-88 standard
S95	ANSI/ISA-95 standard
SOAP	Simple Object Access Protocol
URL	Uniform Resource Locator
XML	Extensible Markup Language
XSD	XML Schema Definition

Integration Touchpoints

The DCS Adapter supports the following touchpoints between an MES and the DCS:

- Create DCS Batch
  Create a new batch in the DCS for a particular DCS master recipe and a set of parameters.
  This is typically triggered from the MES when a specific part of the MES recipe shall be executed automatically by a DCS.
- Get DCS Alarms
  Receive GMP-relevant alarm events from the DCS.
  This is typically done to document the alarms as MES exceptions, hence it uses the review-by-exception features of the MES.
- Get DCS Batch Values
  Read values of the batch report values from the DCS.
  This is typically done to re-use the values in MES processing/calculations or to add them to the MES batch report.
- Set Order Context
  Send information about an MES order including the list of material parameters (input, output, and transfer materials).
  This is typically done from the MES when a specific part of the MES recipe shall be executed automatically by a DCS.
- Set DCS Parameters
  Send DCS parameters to the DCS.
  This is typically done from the MES when a specific part of the MES recipe shall be executed automatically by a DCS.
- Get DCS Parameters
  Read DCS parameters from the DCS.
  This is typically done to re-use the values in MES processing/calculations or to add them to the MES batch report.
- Process Consumed Material
  Receive information about material consumptions (aka goods receipts) performed on the DCS as part of a specific order.
  This is typically done so that the MES can apply the corresponding changes on the inventory and document the consumed materials in the MES batch report.
- Process Produced Material
  Receive information about material production (aka goods issues) performed on the DCS as part of a specific order.
  This is typically done so that the MES can apply the corresponding changes on the inventory and document the produced materials in the MES batch report.

- Get Batch Information
  Read all relevant information of a batch from the MES.
  This is typically done from the DCS to retrieve the batch status and other information relevant for automatic execution.
- Process DCS Event
  A DCS sends a DCS event message to the MES.
  Typically, this is done to synchronize an MES with a DCS.

Architecture of an MES Integrated with the DCS Adapter

The figure below illustrates the architecture of all involved components.

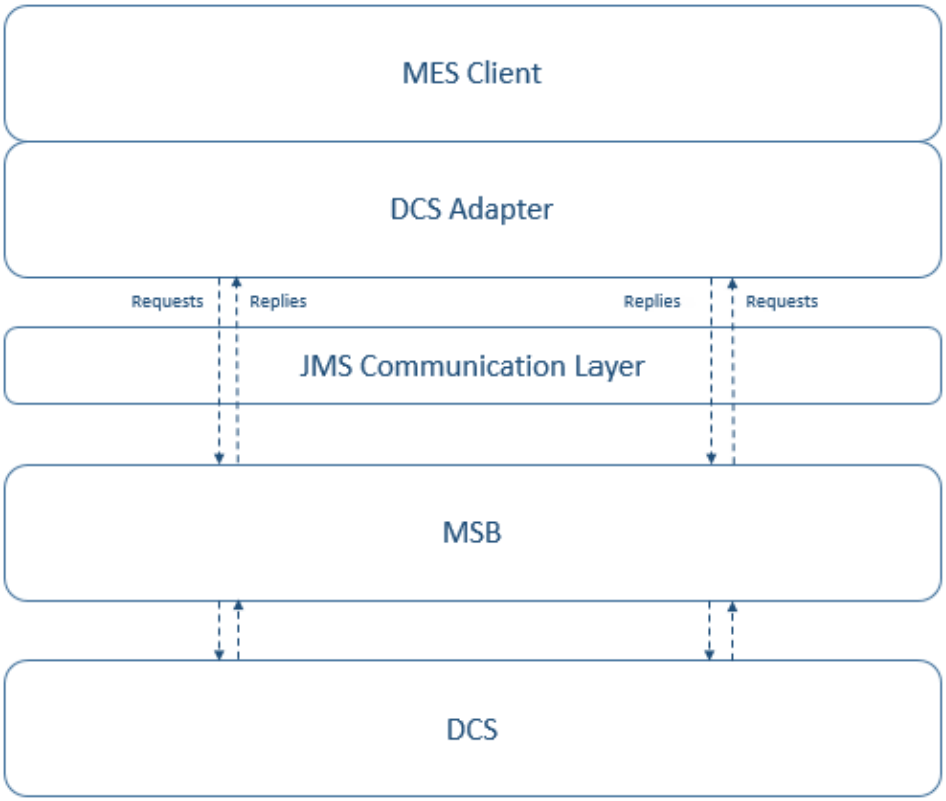

Figure 1: Architecture

The DCS Adapter typically runs within the MES layer. The MES client calls the DCS Adapter to communicate with the DCS. The DCS Adapter creates and sends a request via the Java Message Service (JMS) communication layer.

The Manufacturing Service Bus (MSB) is a middleware component that communicates with the MES via JMS and with a DCS via the DCS-specific technology. The MSB adopts, translates, and routes the defined MES requests to the specific DCS according to their interface specification. For the communication, a DCS can use different technologies, e.g. DCOM or web services.

The communication is bidirectional, i.e. a DCS can also send requests to an MES and receive replies. Again, the MSB adopts, translates, and routes the DCS requests to the DCS Adapter.

The usage of an MSB is not mandatory. For simple integration scenarios, it is sufficient to implement a specific JMS message handler to integrate a particular DCS.

Responsibility of the Components

Each component has its specific responsibility when an MES is connected with a DCS by means of the DCS Adapter.

MES client:

- Provides the parameters as needed by the DCS to execute a request.
- Calls the DCS Adapter.
- Evaluates the reply from the DCS and records the data as needed.
- Defines listeners that process requests coming from a DCS on a high level: they apply the corresponding business logic and return a result.

DCS Adapter:

- Provides an interface to a DCS that is independent of a specific DCS.
- Creates the requests and sends them via JMS.
- Waits for the defined reply and returns the results of the DCS to the MES client.
- Defines listeners that process requests coming from a DCS on a low level: they convert the requests to objects that can be handled by the MES, call the corresponding high-level listeners, and then send the result as a reply to the DCS via JMS.

Manufacturing Service Bus:

- Listens to the requests of all systems connected to it. The sender of a request could be the DCS Adapter or a DCS.
- Translates the received requests into the language of the receiver (a specific DCS or the DCS Adapter) and communicates with the receiver according to the corresponding interface.
- Translates the replies of the receiver into the defined reply message of the sender (a specific DCS or the DCS Adapter) and sends the replies.

Error Handling

The following types of errors can occur:

- JMS broker not available:
  In this case, not even the JMS request can be sent to the MSB.
- Request timeout, typically because the MSB is not available:
  If no component listens to the message sent by the DCS Adapter, the DCS Adapter runs into a timeout.
  This can also happen if the configured timeout is too short. That means the timeout is smaller than the time the DCS or the MES require to process the request.
- An error occurred in the DCS or the MES:
  In this case, the MSB or the MES reply that the request has been rejected. The reply contains a corresponding error message.

A *DCSException* contains a localizable error message and a detailed error message:

- The error message contains general information about the issue.
- The detailed error message contains information about the root cause of the issue.

The component using the DSC Adapter (e.g. an MES client) must handle the exceptions. Usually it displays an error dialog with the error messages.
The DCS Adapter provides the *DCS<SpecificName>Exception* subclasses for each exception (e.g. *DCSCommunicationException*, *DCSJMSNotAvailableException*). Refer to the subclasses if the client shall distinguish between the different error cases.

Chapter 3

Integration into an MES

The integration of the DCS Adapter into an MES requires several steps:

1. Install the DCS Adapter (page 10).
2. Set up the DCS Adapter configuration (page 11).
3. Make a message broker available (page 11).
4. Connect an MES client to a DCS (page 12).
5. Implement the high-level message listeners for all relevant requests coming from a DCS (page 13).
6. Configure the DCS Adapter (page 15).

We recommend to perform the following tasks before using the DCS Adapter in a productive environment:

- Simulate a DCS with a Java mock (page 16).
- Test the MES client without a DCS (page 18).

Prerequisites Checklist

Before you start the installation, check the prerequisites:

	Prerequisite	Your Notes	Done?
1	ActiveMQ is available in version 5.16.3 as required by the DCS Adapter.		

Installing the DCS Adapter

TIP

Starting with FT PharmaSuite 8.4, the DCS Adapter is included in FT PharmaSuite and will be installed along with FT PharmaSuite. However, a DCS Adapter-specific license is required.
In case the DCS Adapter is not installed, proceed as described below.

The DCS Adapter is deployed with following JAR files:

- *dcs-adapter-<version>.jar*
- *dcs-adapter-<version>-javadoc.jar*

TIP
For the exact version number, please refer to section "Introduction" (page 1).

To install the DCS Adapter, proceed as follows:

1. Open Internet Explorer and navigate to the Rockwell Automation Download Site.
2. Navigate to the DCS Adapter and download the package.
3. On the Windows machine, expand the file that you have downloaded to extract the JAR files to a directory of your choice.
4. Add the JAR files to your development environment.
5. Make sure that the ActiveMQ JAR files are available in your development environment.
6. Extend the classpath accordingly.
7. Optional (only if the DCS Adapter is used with FT PharmaSuite):
   For FactoryTalk ProductionCentre-based applications, add the JAR files as **Library** objects in Process Designer.

Setting up the DCS Adapter Configuration

The *dcs-adapter-<version>.jar* file provides properties files that can be used to configure and customize the DCS Adapter:

> **TIP**
> For the exact version number, please refer to section "Introduction" (page 1)

- *ServiceImplementations.properties* allows to assign a custom implementation class to services and bean interfaces used by the DCS Adapter.
- *log4j2.xml* contains the log4j configuration of the DCS Adapter. You can change logging levels and other settings.
- *DCSConfig.properties* contains basic configuration properties of the DCS Adapter. Currently, the file contains only one property: **SystemName**. This is the name of the MES system used by the DCS Adapter. The name is used to automatically fill the sender field for all requests sent by the DCS Adapter and should be used by a DCS to fill the receiver field when sending requests to the MES. By default, **SystemName** is **MES**, but you can change it.
- *DCSAdapterMsgPack.properties* contains all error messages used by the DCS Adapter.

Making a Message Broker Available

In order to use JMS, a message broker is needed. The DCS Adapter supports the following message brokers:

- ActiveMQ broker of FT PharmaSuite
  You can use the message broker of your FT PharmaSuite installation. The broker is installed as an MS Windows service and named **FT PharmaSuite ActiveMQ Broker**. For details, see "FT PharmaSuite Technical Guide Installation" [A1] (page 59).
- ActiveMQ broker of FactoryTalk ProductionCentre
  You can start the broker of your FactoryTalk ProductionCentre installation. The broker is installed as an MS Windows service and named **ActiveMQ**. For details, see "FactoryTalk ProductionCentre Plant Operations Release 10.4 Server Installation Guide - JBoss Advanced" [A2] (page 59).
- A separate ActiveMQ broker.

Connecting an MES Client to a DCS

The interface of the DCS Adapter is provided as a Java service with methods for each integration touchpoint (page 3) to the DCS. For details, see the Java documentation of a service.

The tasks of the MES client are:

- call the service methods and provide the parameters as needed,
- evaluate the reply to the service call and record the data as needed.

The *IDCSService* is responsible for the tasks. It will be instantiated by using a factory:

```
import com.rockwell.mes.dcs.ifc.IDCSService;
...
IDCSService service = com.rockwell.mes.dcs.ifc.DCSServiceFactory.getDCSService();
```

Example for the create DCS batch touchpoint

```
import com.rockwell.mes.dcs.ifc.IDCSService;
...
IDCSService service = com.rockwell.mes.dcs.ifc.DCSServiceFactory.getDCSService();

IDCSBatchCreationParameter params = new DCSBatchCreationParameter("MyRecipeID");
params.setCampaignID("MyCampaignID").setFormulaID("MyFormulaID") ...

IDCSAdapterConfiguration config = new DCSAdapterConfiguration(brokerURL, timeout);

service.createDCSBatch(config, "MyDCS", "NewbatchID", batchCreationParameter);
```

For details about the configuration, see "Configuring the DCS Adapter" (page 15).

Implementing Listeners

The DCS Adapter supports a bidirectional communication between a DCS and an MES: either an MES sends the request and a DCS replies or a DCS sends the request and an MES replies. However, when the **ProcessDCSEvent** request (page 42) is used in the asynchronous use case, no reply is sent.
Please refer to "Connecting an MES Client to a DCS" (page 12) for details how to invoke an **MES sender** communication. To invoke a **DCS sender** communication, implement a listener for each relevant integration touchpoint to the DCS (page 3).

The DCS Adapter provides the low-level listeners for all integration touchpoints as abstract classes that extend the *AbstractDCSMessageListener* class. The low-level listeners convert the JMS message coming from a DCS to a bean object that holds all relevant information of the request. Additionally, the low-level listeners are generic in regards to the sent/received message types and the bean objects created by them. They are based on the requests and provide a single abstract method that should be implemented on the higher level:

Abstract method to be implemented on a higher level

```
/**
   * Process the request.
   *
   * @param sender the name of the DCS system that is sending the request
   * @param dcsObject the DCS object containing the request information
   * @param otherInformations the other informations that came with the request as a map
   * of key/value pairs. Supported Objects of the value are String, BigDecimal, Integer,
   * Calendar and Boolean
   * @return the response
   */
  protected abstract IDCSResponse processRequest(final String sender, final ReceivedType
            dcsObject, final Map<String, Object> otherInformations);
```

The returned response should wrap an **IResultContainer** object that contains the result required by this listener. Each listener contains a *createExtendedResult(Map<String, Object>)* method that can be used to create a result object of the correct type. For details, see the Java documentation of the listeners.
The low-level listener for the *Process DCS Event* message sends the reply only in case of synchronous events.

Example implementation of a listener that handles batch information requests coming from a DCS

```
public class MyGetBatchInformationMessageListener extends
             GetBatchInformationMessageListener {
  /**
   * @param dcsAdapterConfig the DCS adapter configuration
   */
  public MyGetBatchInformationMessageListener(IDCSAdapterConfiguration
         dcsAdapterConfig) {
    super(dcsAdapterConfig,
          DCSConfiguration.INSTANCE.getStringValue(DCSConfiguration.SYSTEM_NAME));
  }

  @Override
  protected IDCSResponse processRequest(final String sender, IBatchInformationParameter
            dcsObject, final Map<String, Object> otherInformations) {
    IDCSResponse<IExtendedBatchInformationResult> response =
                                      newDCSResponse<IExtendedBatchInformationResult>();
    IBatchInformationResult specificResult =
                DCSServiceFactory.createBean(IBatchInformationResult.class);
    String batchID = dcsObject.getBatchID();
    Batch batch = PCContext.getFunctions().getBatchByName(batchID);
    if (batch == null) {
      response.setReplySuccessful(false);
      response.setReplyErrorMessage("Batch does not exist.");
      specificResult = null;
    } else {
     specificResult.setBatchID(batch.getName());
     specificResult.setMaterialID(batch.getPart());
     ... // set all other fields of the result here
     response.setReplySuccessful(true);
    }
    // if you want to send other information in the reply,
    // add it in the map of the extended result
    IExtendedBatchInformationResult result = createExtendedResult(specificResult,
                  Collections.EMPTY_MAP);
    response.setResult(result);
    return response;
  }
```

After implementing the listener, you must create an instance of it and start it using the s*tartReceiving()* method. For each request from a DCS, the *processRequest* method is invoked.

TIP

When you have finished the usage of the listener, do not forget to call its *shutdown()* method to clean up the resources that were used by the listener.

You may have noticed that the Java service of the DCS Adapter contains also methods that allow to send requests for the integration touchpoints where it is usually expected that a DCS sends requests to an MES. This is done for the following reasons: it allows you to test your listeners without using a real DCS and it allows an MES to simulate requests being sent by a DCS, e.g. to add additional consumptions.

Usually, there is one listener per touchpoint, which is responsible for processing all messages of a touchpoint. However, the listener for the *Process DCS Event* messages is intended to run in a phase building block of FT PharmaSuite within an order. This kind of listener shall only process messages that are specific to the batch and material of the order of the phase building block and the event identifier that is configured within the phase building block. It depends on the recipe and the number of running order how many multiple listeners are running at the same time. Each listener is interested in a different message. Therefore the constructor of *AbstractMessageEventMessageListener* has additional parameters in order to get only the corresponding messages.

Example snippet implementation of a DCS event listener that handles DCS event requests coming from a DCS

```
public class MyMessageEventMessageListener extends
             AbstractMessageEventMessageListener {
  /**
   * @param dcsAdapterConfig the DCS adapter configuration
   * @param batchID the not empty batch ID the listener will listen on
   * @param materialID the not empty material ID the listener will listen on
   * @param identifier the not empty event identifier the listener will listen on
  */
  public MyMessageEventMessageListener (IDCSAdapterConfiguration
         dcsAdapterConfig,
         final String batchID, final String materialID, String identifier) {
    super(dcsAdapterConfig,
          DCSConfiguration.INSTANCE.getStringValue(DCSConfiguration.SYSTEM_NAME),
          batchID, materialID, identifier);
  }

  ...
```

Configuring the DCS Adapter

The DCS Adapter expects its configuration as a parameter at each service method. The configuration is bundled in a class implementing the *IDCSAdapterConfiguration* interface. The following parameters are needed:

Parameter	Description	Example
MessageBrokerURL	URL of the message broker.	ssl://hostname:61647
MessageTimeoutInSeconds	Time in seconds needed to send the request.	2

Parameter	Description	Example
MessageTimeToLive	Specifies in milliseconds how long the message system shall retain a message after it has been dispatched and determines how long the system shall wait for a reply message, if one is expected. Default setting = 5 min, Unlimited time = 0	300000

TIP

To prevent the JMS queue from becoming very large or even full, we highly recommend to specify the expiration of the DCS Event JMS message.

Simulating a DCS with a Java Mock

To test the connection of an MES client to a DCS or to test requests that are normally sent by a DCS and processed by MES, a mock for the DCS can be used instead of a real DCS. The DCS Adapter provides a corresponding Java mock that can be started with a test form (see section "Testing the MES Client without a DCS" (page 18)).

To test the connection of an MES to a DCS, the mock acts as a DCS. The response of the mock is controlled by specific input of the parameters.

Request	Reply
Create DCS Batch	■ If *BatchID* starts with *Bad*, the reply is not successful, otherwise the reply is successful.
Get DCS Alarms	■ If *BatchID* starts with *Bad*, the reply is not successful, otherwise the reply is successful ■ Filter criteria are implemented on a fixed set of alarms.
Get DCS Batch Values	■ If *BatchID* starts with *Bad*, the reply is not successful, otherwise the reply is successful. ■ For each requested batch report value, a value is replied. The type depends on the specification.
Set Order Context	■ If *OrderID* starts with *Bad*, the reply is not successful, otherwise the reply is successful.

For testing the connection of a simulated DCS to the MES two options are available: either testing is performed with a real MES or testing with the DCS Adapter includes also the simulation of the MES.

The mock provides listeners that act as an MES. This is useful in case an MES is simulated as well.
To test a real MES, the listeners are not needed since the real MES provides the listeners. However, the listeners of the mock must be stopped.
The response of the mock listener (aka the simulated MES) is controlled by specific input of the parameters.

Request	Reply
Process Consumed Material	■ If *OrderID* starts with *Bad*, the reply is not successful, otherwise the reply is successful.
Process Produced Material	■ If *OrderID* starts with *Bad*, the reply is not successful, otherwise the reply is successful.
Get Batch Information	■ If *BatchID* starts with *Bad*, the reply is not successful, otherwise the reply is successful. ■ The returned batch status depends on the *BatchID*: if it starts with *Released*, *Quarantined*, or *Blocked*, the corresponding status is returned. If it starts with anything else, *OTHER* is returned as batch status. The other batch information fields are filled with fixed values.
Process DCS Event	■ Listen for a given batchID = "BX123", materialID = "Water", and identifier "identifier01". ■ The mock listener alternately returns a successful reply and an error reply.

The mock can be parameterized with the following parameters:

Parameter	Description	Example
MessageBrokerURL	URL of the message broker.	ssl://localhost:61647
Receiver	The mock handles all requests sent to the given receiver/DCS.	JavaDCSMock

TIP

Running the Java mock within the **TestDCSAdapter** tool (page 18) uses the default value for the receiver.

The listeners of the *Process Consumed Material*, *Process Produced Material*, *Get Batch Information*, and *Process DCS Event* requests use MES by default as a receiver name, because they are intended to simulate the high-level listeners implemented by an MES.

Testing the MES Client without a DCS

To test the DCS integration without a real DCS, the **TestDCSAdapter** tool provided for this purpose can be used.

The message type defines who simulates the MES and the DCS in the **TestDCSAdapter** tool:

Message Type	TestDCSAdapter	Java DCS Mock
MES » DCS	Simulates the MES	Simulates the DCS
DCS » MES	Simulates the DCS	Simulates the MES

Depending on whether or not the Java DCS Mock is running and on the message type, you can either test a real DCS or a real MES.

To run the **TestDCSAdapter** tool, proceed as follows:

1. Navigate to the *dcs-adapter-<version>.jar* file.

   TIP
   For the exact version number, please refer to section "Introduction" (page 1)

2. Start the *com.rockwell.mes.dcs.impl.ui.TestDCSAdapter* Java main as an **Application**.

Figure 2: TestDCSAdapter tool

3. The upper part of the form contains the configuration of the DCS Adapter. Here you can configure the message broker URL and start the Java DCS simulation mock (page 16).
   The mock handles messages sent to the **JavaDCSMock** receiver. If your MES client uses another broker URL, adapt it here before you start the mock.
   For message types that are sent by the DCS to the MES, the mock simulates the MES listening to messages from the DCS and the TestDCSAdapter simulates the DCS sending messages to the MES.
   In order to test the MES in that case, you must not start the mock or stop it first.
   For each channel, a listener is running in the mock. Each listener is independent from the other listeners, for example, the **SetDCSParameters** request does not set DCS parameters on the *GetDCSParameters* listener. For listeners that return result values, specific tabs are available that allow to inject values in the corresponding listeners.

4. To start the mock, click the **Start Java DCS Mock** button. After the mock has been started, it can send replies to the MES client or to the TestDCSAdapter tool.

   - In this context only, the *Add Alarm Events* tab is useful. Here you can dynamically add alarm events to the running mock on the fly. To define the alarm events, use the input boxes. The alarm event will get the current timestamp. The next time a request to get the alarm events is replied to, the newly added alarm events are included (depending on the configured batch, unit, modules).

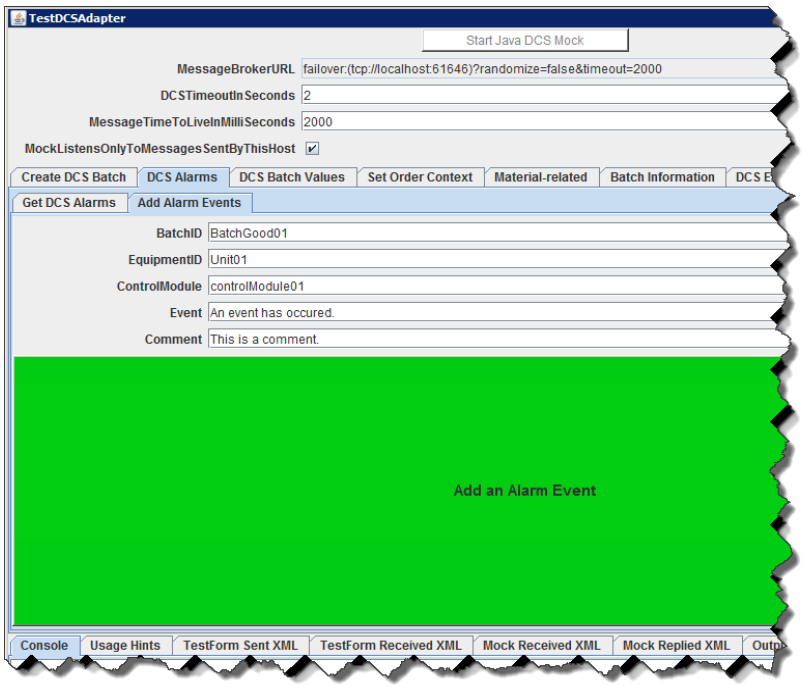

Figure 3: Adding alarm events to the mock

- In the *Set DCS Batch Values* tab, you can dynamically add batch values that can later be retrieved by the *Get DCS Batch Values* tab or any phase building block of FT PharmaSuite that reads values from a DCS.
  The **dateTime** type is not supported. If you request a **dateTime** value in the *Get DCS Batch Values* tab, the current time is reported.

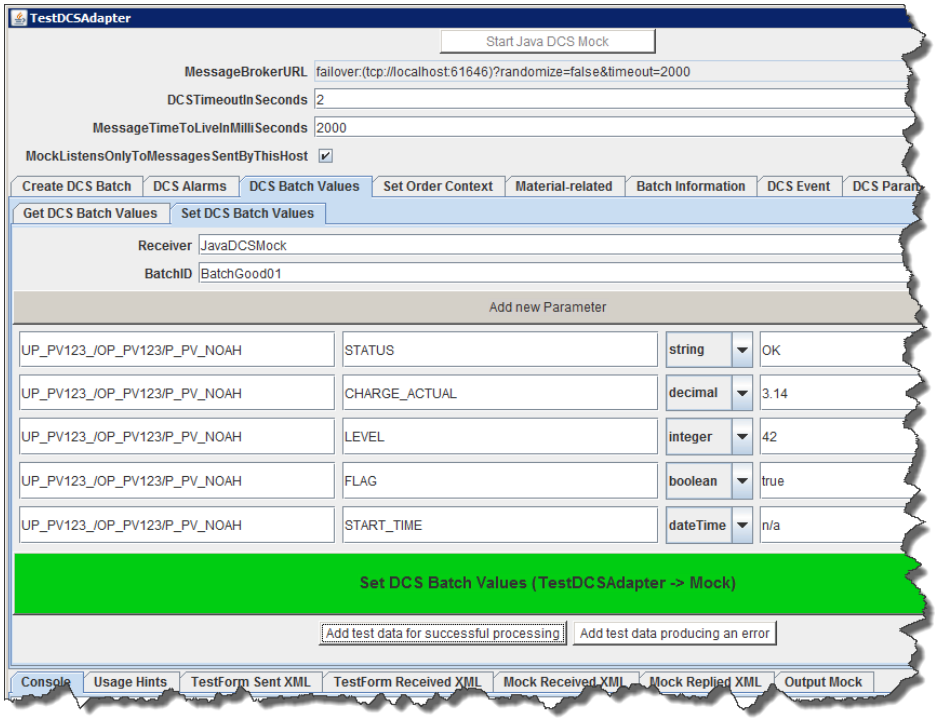

Figure 4: Adding batch values to the mock

- In the *Set DCS Parameter to Mock* tab, you can dynamically add DCS parameter values which can be retrieved later by the *Get DCS Parameter* tab or any phase building block of FT PharmaSuite that reads values from a DCS.
  The **dateTime** type is not supported. If you request a **dateTime** value in the *Get DCS Parameter* tab, the current time is reported.

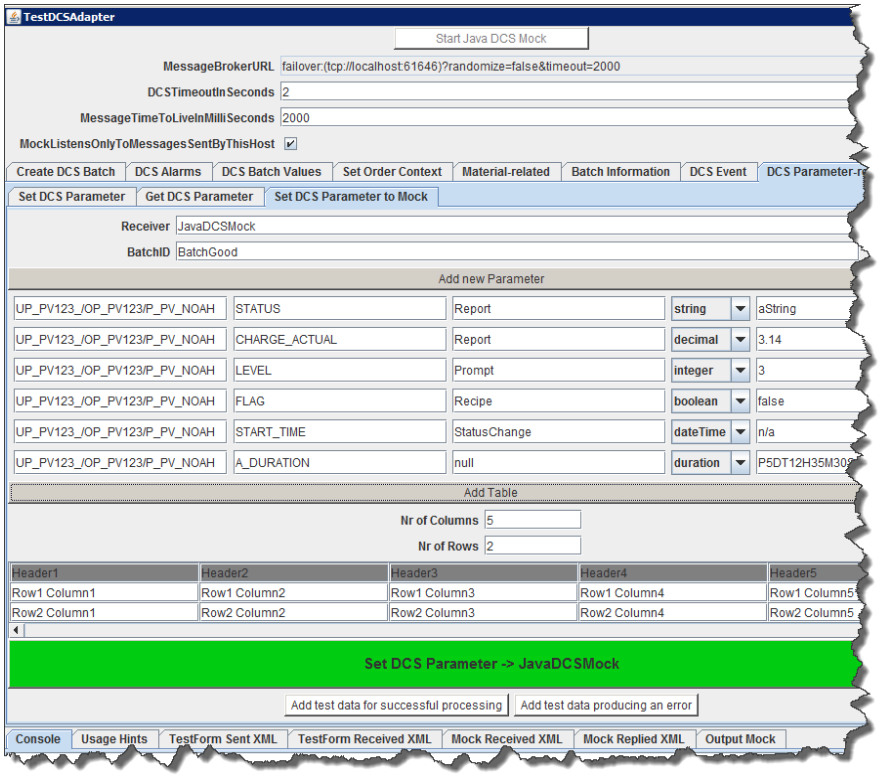

Figure 5: Adding DCS parameters to the mock

5. You can use the *Batch Information* tab, the two sub-tabs of the *Material-related* tab, and the *DCS Event* tab to test the DCS integration (see "Testing the DCS Integration without an MES Client" (page 53)) or to test the high-level listeners implemented by an MES or a phase building block of FT PharmaSuite. By default, a listener mock is started for each of the tabs. This allows you to test the DCS integration. However, if you wish to test a high-level listener, you can stop the mock listener with the **Stop listener** button in the corresponding tab. Then all messages are processed by the running MES listener.
   In both tabs, requests are sent that would normally be sent by a DCS. This is different from the other tabs where requests are sent that are normally sent by an MES.

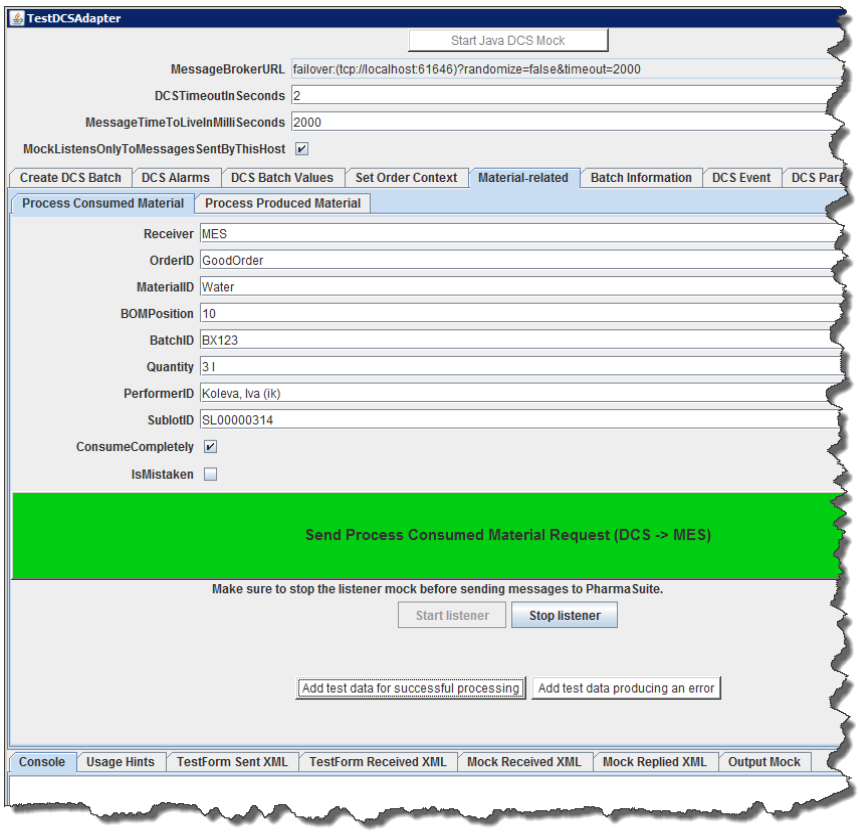

Figure 6: Stopping and starting listener mocks

6. The other tabs of the form are mainly intended for testing the DCS integration and to simulate an MES (see section "Testing the DCS Integration without an MES Client" (page 53)).

Chapter 4

Integration of a DCS

To integrate a DCS, a component must be created that listens to the DCS requests of the MES for the given DCS and translates it into the language of the specific DCS. Furthermore, for requests sent by the DCS to the MES, the component must translate them into the language of the MES. For integrating several DCSs, it makes sense to create an MSB that communicates with multiple DCSs according to their technology.
For simple integration scenarios, you can also implement a specific JMS message handler to integrate a specific DCS.

For details of the tasks of an MSB, see section "Responsibility of the Components" (page 6).

The integration of a DCS makes use of:

- Channels for integration touchpoints (page 24)
- XML schema of the interface (page 25)
- Touchpoint-specific requests:
  - ProcessCreateDCSBatch (page 27)
  - GetDCSAlarmEvents (page 29)
  - GetDCSBatchValues (page 31)
  - ProcessOrderContext (page 34)
  - ProcessConsumedMaterial (page 36)
  - ProcessProducedMaterial (page 38)
  - GetBatchInformation (page 40)
  - ProcessDCSEvent (page 42)
  - ProcessDCSParameters (page 44)
  - GetDCSParameters (page 47)
- Touchpoint-specific replies
  - AcknowledgeCreateDCSBatch (page 28)
  - ShowDCSAlarmEvents (page 30)
  - ShowDCSBatchValues (page 33)
  - AcknowledgeOrderContext (page 35)

- AcknowledgeConsumedMaterial (page 37)
- AcknowledgeProducedMaterial (page 39)
- ShowBatchInformation (page 41)
- AcknowledgeDCSEvent (page 43)
- AcknowledgeDCSParameters (page 46)
- ShowDCSParameters (page 48)

The example of an MSB implementation based on EIHub (page 51) describes how to structure such an integration flow.

We recommend to perform the following task before using the DCS Adapter in a productive environment:

- Test the DCS integration without an MES client (page 53).

Channels for Integration Touchpoints

For each DCS, the following channels are used, i.e. one channel per integration touch point:

Integration touchpoint	Channel
Create DCS Batch	DCSRequest_<DCSTargetSystem>_CreateDCSBatch
Get DCS Alarms	DCSRequest_<DCSTargetSystem>_GetDCSAlarmEvents
Get DCS Batch Values	DCSRequest_<DCSTargetSystem>_GetDCSBatchValues
Set Order Context	DCSRequest_<DCSTargetSystem>_ProcessOrderContext
Set DCS Parameters	DCSRequest_<DCSTargetSystem>_ProcessDCSParameters
Get DCS Parameters	DCSRequest_<DCSTargetSystem>_GetDCSParameters

Additionally, four channels are used for the messages received by an MES:

Integration touchpoint	Channel
Process Consumed Material	DCSRequest_<MESSystemName>_ProcessConsumedMaterial
Process Produced Material	DCSRequest_<MESSystemName>_ProcessProducedMaterial
Get Batch Information	DCSRequest_<MESSystemName>_GetBatchInformation
Process DCS Event	DCSRequest_<MESSystemName>_ProcessDCSEvent

The common pattern of a channel is:
DCSRequest_<LogicalReceiverName>_<RequestType>

TIP

We highly recommend to handle the reply of requests sent by a DCS to an MES in the MSB. This is important to increase the detectability of lost messages.

When the **ProcessDCSEvent** request is used in the asynchronous use case, no reply is sent.

XML Schema of the Interface

The interface between the DCS Adapter and the DCS or MSB is an XML document sent via JMS.

TIP

For the exact version number, please refer to section "Introduction" (page 1)

The corresponding XML schema is defined in *mes-dcs-interface.xsd* and provided with the *dcs-adapter-<version>.jar*. The schema is compliant to the S88/S95 standards and based on the B2MML standard schema.

For each integration touchpoint, the schema definition contains two top-level XML elements, one for the request and one for the reply:

Top-level XML element for requests	Top-level XML element for replies
ProcessCreateDCSBatch (page 27)	AcknowledgeCreateDCSBatch (page 28)
GetDCSAlarmEvents (page 29)	ShowDCSAlarmEvents (page 30)
GetDCSBatchValues (page 31)	ShowDCSBatchValues (page 33)
ProcessOrderContext (page 34)	AcknowledgeOrderContext (page 35)
ProcessConsumedMaterial (page 36)	AcknowledgeConsumedMaterial (page 37)
ProcessProducedMaterial (page 38)	AcknowledgeProducedMaterial (page 39)
GetBatchInformation (page 40)	ShowBatchInformation (page 41)
ProcessDCSEvent (page 42)	AcknowledgeDCSEvent (page 43)
ProcessDCSParameters (page 44)	AcknowledgeDCSParameters (page 46)
GetDCSParameters (page 47)	ShowDCSParameters (page 48)

The DCS Adapter sends an XML request of a specific document type (e.g. **ProcessCreateDCSBatch**) and expects a reply of the corresponding document type (e.g. **AcknowledgeCreateDCSBatch**).

The messages in both directions are text messages containing XML-formatted strings.

TIP

To validate the sample XML documents listed for the requests and replies, move the files to the *../../../main/xsd/mes-dcs-interface.xsd* subdirectory.
We highly recommend to handle the reply of requests sent by a DCS to an MES in the MSB. This is important to detect whether the MES has successfully processed the message or not.

General Structure of the XML Requests and Replies

In general, each XML request and reply consists of three containers:

- **ApplicationArea** contains the sender and the receiver of a message.
- **DataArea**
  - For requests, it contains the parameters of the request.
  - For replies, it contains the information whether the request has been accepted or rejected and an error message in the latter case.
- **OtherInformations** is a container that can be used to transfer additional data that is needed and not contained in the standard. Optionally, it contains a list of additional key/value pairs (see section "Extending the Standard" (page 55)).

Request: ProcessCreateDCSBatch

The request for creation of a new DCS batch is an XML document of the **ProcessCreateDCSBatch** type. The parameters are located in the **DataArea** container (page 26):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **ControlRecipe/Parameters** contains the list of any additional parameters.
- **ControlRecipeEquipmentRequirements** contains the list of the required units.

Example of a 'create DCS batch' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ProcessCreateDCSBatch
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>Building_API_220</Receiver>
  </ApplicationArea>

  <DataArea>
    <BatchID>BS101338_27Oct15_2111_PV51400</BatchID>
    <MasterRecipeID>UP_BUFP_BUFFER_MAKEUP_MP</MasterRecipeID>
    <ControlRecipe>
      <Parameters>
        <Parameter>
          <ID>R_BUF_VOL</ID>
          <ValueNumeric>765</ValueNumeric>
        </Parameter>
        <Parameter>
          <ID>ENCODER</ID>
          <ValueString>04PV51400ENC1</ValueString>
        </Parameter>
        <Parameter>
          <ID>R_INT_WIFI_RINSE</ID>
          <ValueBoolean>true</ValueBoolean>
        </Parameter>
      </Parameters>
      <EquipmentRequirements>
        <EquipmentRequirement>
          <Constraint>UnitProcA</Constraint>
          <ID>PV3553</ID>
        </EquipmentRequirement>
        <EquipmentRequirement>
          <Constraint>EqmClassX</Constraint>
          <ID>PV4253</ID>
        </EquipmentRequirement>
      </EquipmentRequirements>
    </ControlRecipe>

    <!-- all optional fields filled -->
    <CampaignID>DeltaV_V11_CAMP</CampaignID>
```

```
    <ScaledSize>100.00</ScaledSize>
    <FormulaID>SFPHC_PV515400_MAKEUP</FormulaID>
    <Description>This is a long description</Description>
  </DataArea>
</ProcessCreateDCSBatch>
```

Reply: AcknowledgeCreateDCSBatch

The reply to the **ProcessCreateDCSBatch** request is an XML document of the **AcknowledgeCreateDCSBatch** type. The results are located in the **DataArea** container (page 26):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/InternalBatchID** contains the unique internal ID of the newly created batch on the DCS.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.

Example of an Accepted reply to a 'create DCS batch' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeCreateDCSBatch
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>
    <InternalBatchID>batchID_20160819_104003</InternalBatchID>
  </DataArea>

</AcknowledgeCreateDCSBatch>
```

Example of a Rejected reply to a 'create DCS batch' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeCreateDCSBatch
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected">
      Error during batch creation. Batch ID already exists.
```

```
    </ResponseCriteria>
    <InternalBatchID></InternalBatchID>
  </DataArea>

</AcknowledgeCreateDCSBatch>
```

Request: GetDCSAlarmEvents

The request for getting alarm events from a DCS is an XML document of the **GetDCSAlarmEvents** type. The parameters are located in the **DataArea** container (page 26):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **DataArea/EquipmentID** contains the identifier of the unit.
- **DataArea/RecipeElement** contains the list of control modules.

Example of a 'get DCS alarm events' request

```
<?xml version="1.0" encoding="UTF-8"?>

<GetDCSAlarmEvents
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <!-- Maximal filter criteria: use all fields -->
      <BatchID>BX01815-20160114_142303_unitA</BatchID>
    <EquipmentID>VesselA</EquipmentID>
      <RecipeElement>
        <ActualEquipmentID>controlModuleA</ActualEquipmentID>
        <ActualEquipmentID>controlModuleB</ActualEquipmentID>
        <ActualEquipmentID>controlModuleC</ActualEquipmentID>
      </RecipeElement>
    <StartTime>2016-01-14T13:45:00</StartTime>
    <EndTime>2016-01-14T17:45:00</EndTime>
  </DataArea>

</GetDCSAlarmEvents>
```

Reply: ShowDCSAlarmEvents

The reply to the **GetDCSAlarmsEvents** request is an XML document of the **ShowDCSAlarmEvents** type. The results are located in the **DataArea** container (page 26):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/Events** contains the returned list of alarm events if **actionCode** contains **Accepted**. The list may be empty.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.

Example of an Accepted reply to a 'get DCS alarm events' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ShowDCSAlarmEvents
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>
    <Events>
      <AlarmEvent>
        <TimeStamp>2016-01-14T13:45:34</TimeStamp>
        <Value>Errorcode:3;Critical;Upper limit violation</Value>
        <EquipmentID>VesselA/TempSensor</EquipmentID>
        <MessageText>An arbitrary comment.</MessageText>
      </AlarmEvent>
      <AlarmEvent>
        <TimeStamp>2016-01-14T14:44:14</TimeStamp>
        <Value>Errorcode:4;Critical;Lower limit violation</Value>
        <EquipmentID>VesselA/TempSensor</EquipmentID>
      </AlarmEvent>
    </Events>
  </DataArea>

</ShowDCSAlarmEvents>
```

Example of a Rejected reply to a 'get DCS alarm events' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ShowDCSAlarmEvents
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected”>
      An error has been occurred.
    </ResponseCriteria>
    <Events>       </Events>
  </DataArea>

</ShowDCSAlarmEvents>
```

Request: GetDCSBatchValues

The request for getting batch values from a DCS is an XML document of the **GetDCSABatchValues** type. The parameters are located in the **DataArea** container (page 26):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **DataArea/ControlRecipe** contains the list of requested batch values:
  - **RecipeElement/ID** contains the path of the batch value.
  - **RecipeElement/ParameterID** contains the name of the batch value.
  - **RecipeElement/DataType** contains the requested data type of the batch value.

Example of a 'get DCS batch values' request

```
<?xml version="1.0" encoding="UTF-8"?>

<GetDCSBatchValues
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <BatchID>BX01815-20160114_142303_unitA</BatchID>

    <ControlRecipe>
      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>STATUS</ParameterID>
        <DataType>string</DataType>
      </RecipeElement>
      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>FLAG</ParameterID>
        <DataType>boolean</DataType>
      </RecipeElement>
      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>CHARGE_ACTUAL</ParameterID>
        <DataType>decimal</DataType>
      </RecipeElement>
      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>START_TIME</ParameterID>
        <DataType>dateTime</DataType>
      </RecipeElement>
    </ControlRecipe>
  </DataArea>

</GetDCSBatchValues>
```

Reply: ShowDCSBatchValues

The reply to the **GetDCSBatchValues** request is an XML document of the **ShowDCSBatchValues** type. The results are located in the **DataArea** container (page 26):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ControlRecipe** contains the list of returned batch values:
  - **RecipeElement/ID** contains the path of the batch value.
  - **RecipeElement/ParameterID** contains the name of the batch value.
  - **RecipeElement/Value<data type>** contains the returned batch value of the requested data type.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.

Example of an Accepted reply to a 'get DCS batch values' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ShowDCSBatchValues
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>

    <ControlRecipe>
      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>STATUS</ParameterID>
        <ValueString>started</ValueString>
      </RecipeElement>

      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>START_TIME</ParameterID>
        <ValueDatetime>2016-01-14T14:54:43</ValueDatetime>
      </RecipeElement>

      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
        <ParameterID>FLAG</ParameterID>
        <ValueBoolean>true</ValueBoolean>
      </RecipeElement>

      <RecipeElement>
        <ID>UP_PV123_/OP_PV123/P_PV_NOAH</ID>
```

```
        <ParameterID>LEVEL</ParameterID>
        <ValueInteger>12</ValueInteger>
      </RecipeElement>

    </ControlRecipe>
  </DataArea>

</ShowDCSBatchValues>
```

Request: ProcessOrderContext

The request for setting an order context in a DCS is an XML document of the **ProcessOrderContext** type. The order information is located in the **DataArea** container (page 26):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **DataArea/MaterialParameters** contains the list of material parameters for this order.

Example of a 'set order context' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ProcessOrderContext
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>
  <DataArea>
    <OrderID>GoodOrder</OrderID>
    <MaterialDefinitionID>Shiny pills</MaterialDefinitionID>
    <BatchID>BS101338_10Jan16_2111_PV51400</BatchID>
    <MasterRecipeID>UP_BUFP_BUFFER_MAKEUP_MP</MasterRecipeID>
    <MasterRecipeVersion>1</MasterRecipeVersion>
    <PhaseID>Set Order Context (RS) [1.0]</PhaseID>
    <MaterialParameters>
      <Parameter>
        <MaterialDefinitionID>Water</MaterialDefinitionID>
        <MaterialPositionID>10</MaterialPositionID>
        <ParameterType>ProcessInput</ParameterType>
        <Value>
          <Quantity>42</Quantity>
          <UnitOfMeasure>l</UnitOfMeasure>
        </Value>
        <AllocatedBatchID>BX01</AllocatedBatchID>
        <AllocatedBatchID>BX185</AllocatedBatchID>
        <OrderStepID>Dispense</OrderStepID>
      </Parameter>
      <Parameter>
        <MaterialDefinitionID>Shiny pills</MaterialDefinitionID>
        <MaterialPositionID>20</MaterialPositionID>
```

```
        <ParameterType>ProcessOutput</ParameterType>
        <Value>
          <Quantity>10</Quantity>
          <UnitOfMeasure>kg</UnitOfMeasure>
        </Value>
        <OrderStepID>Packaging</OrderStepID>
      </Parameter>
    </MaterialParameters>
  </DataArea>
</ProcessOrderContext>
```

Reply: AcknowledgeOrderContext

The reply to the **ProcessOrderContext** request is an XML document of the **AcknowledgeOrderContext** type. The results are located in the **DataArea** container (page 26):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.

Example of an Accepted reply to a 'set order context' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeOrderContext
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>
  </DataArea>

</AcknowledgeOrderContext>
```

Example of a Rejected reply to a 'set order context' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeOrderContext
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected”>
      An error has been occurred.
    </ResponseCriteria>
  </DataArea>

</AcknowledgeOrderContext >
```

Request: ProcessConsumedMaterial

The request for processing consumed material in an MES is an XML document of the **ProcessConsumedMaterial** type. The information of the consumed material is located in the **DataArea** container (page 26):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **DataArea/doFixMistake** indicates whether the message is related to a previously sent request that was sent by mistake. This field is intended for usage from MES (as part of an error handling mechanism). Any request coming from the automation system that has this field set to **true** will be rejected by MES. If the field is empty, **false** is used by default (i.e. straightforward consumption).
- **DataArea/ConsumeMaterialEvent** contains additional information of the consumed material.

Example of a 'process consumed material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ProcessConsumedMaterial
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>
  <DataArea>
    <doFixMistake>false</doFixMistake>
    <OrderID>GoodOrder</OrderID>
    <MaterialDefinitionID>Water</MaterialDefinitionID>
    <MaterialPositionID>10</MaterialPositionID>
    <BatchID>BX123</BatchID>
    <ConsumeMaterialEvent>
      <TimeStamp>2017-08-15T13:37:41.157+02:00</TimeStamp>
      <Value>
        <Quantity>3</Quantity>
        <UnitOfMeasure>l</UnitOfMeasure>
      </Value>
      <SublotID>SL00000314</SublotID>
      <SublotIsEmpty>true</SublotIsEmpty>
      <PersonID>Gerr, Detamana (dg)</PersonID>
    </ConsumeMaterialEvent>
  </DataArea>
</ProcessConsumedMaterial>
```

Reply: AcknowledgeConsumedMaterial

The reply to the **ProcessConsumedMaterial** request is an XML document of the **AcknowledgeConsumedMaterial** type. The results are located in the **DataArea** container (page 26):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**. The error messages supported by MES are located in the *DCSAdapterMsgPack.properties* file.

Example of an Accepted reply to a 'process consumed material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeConsumedMaterial
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>
  </DataArea>

</AcknowledgeConsumedMaterial>
```

Example of a Rejected reply to a 'process consumed material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeConsumedMaterial
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected”>
      Could not find a matching batch.
    </ResponseCriteria>
  </DataArea>

</AcknowledgeConsumedMaterial>
```

Request: ProcessProducedMaterial

The request for processing produced material in an MES is an XML document of the **ProcessProducedMaterial** type. The information of the produced material is located in the **DataArea** container (page 26):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **DataArea/ProduceMaterialEvent** contains additional information of the produced material.

Example of a 'process produced material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ProcessProducedMaterial
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>
  <DataArea>
    <OrderID>GoodOrder</OrderID>
    <MaterialDefinitionID>Shiny pills</MaterialDefinitionID>
    <MaterialPositionID>110</MaterialPositionID>
    <BatchID>BX123</BatchID>
    <ProduceMaterialEvent>
      <TimeStamp>2017-08-15T13:48:18.807+02:00</TimeStamp>
      <Value>
        <Quantity>10</Quantity>
        <UnitOfMeasure>kg</UnitOfMeasure>
      </Value>
      <SublotID>SL00000315</SublotID>
      <PersonID>Gerr, Detamana (dg)</PersonID>
    </ProduceMaterialEvent>
  </DataArea>
</ProcessProducedMaterial>
```

Reply: AcknowledgeProducedMaterial

The reply to the **ProcessProducedMaterial** request is an XML document of the **AcknowledgeProducedMaterial** type. The results are located in the **DataArea** container (page 26):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**. The error messages supported by MES are located in the *DCSAdapterMsgPack.properties* file.

Example of an Accepted reply to a 'process produced material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeProducedMaterial
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"></ResponseCriteria>
  </DataArea>

</AcknowledgeProducedMaterial>
```

Example of a Rejected reply to a 'process produced material' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeProducedMaterial
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected”>
      The corresponding order is not running.
    </ResponseCriteria>
  </DataArea>

</AcknowledgeProducedMaterial>
```

Request: GetBatchInformation

The request for retrieving batch information from an MES is an XML document of the **GetBatchInformation** type. The batch information is located in the **DataArea** container (page 26):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.

Example of a 'get batch information' request

```
<?xml version="1.0" encoding="UTF-8"?>

<GetBatchInformation
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>
  <DataArea>
    <BatchID>ReleasedBatch</BatchID>
    <MaterialDefinitionID>Shiny pills</MaterialDefinitionID>
 </DataArea>
</GetBatchInformation>
```

Reply: ShowBatchInformation

The reply to the **GetBatchInformation** request is an XML document of the **ShowBatchInformation** type. The results are located in the **DataArea** container (page 26):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.
- **DataArea/BatchStatus** contains an enumeration value representing the batch status.

Example of an Accepted reply to a 'get batch information' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ShowBatchInformation
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"/>
    <BatchID>ReleasedBatch</BatchID>
    <MaterialDefinitionID>Shiny pills</MaterialDefinitionID>
    <ifc:BatchStatus>Released</ifc:BatchStatus>
    <BatchQuantity>
      <Quantity>1000.0</Quantity>
      <UnitOfMeasure>kg</UnitOfMeasure>
    </BatchQuantity>
    <Potency>
```

```
      <ifc:Quantity>87.00</ifc:Quantity>
      <ifc:UnitOfMeasure>%</ifc:UnitOfMeasure>
    </Potency>
    <ProductionDate>2017-08-14T14:10:12.666+02:00</ProductionDate>
    <ExpiryDate>2018-08-15T14:10:12.666+02:00</ExpiryDate>
    <RetestDate>2017-09-15T14:10:12.666+02:00</RetestDate>
  </DataArea>

</ShowBatchInformation>
```

Example of a Rejected reply to a 'get batch information' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ShowBatchInformation
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected”>
      An error has been occurred.
    </ResponseCriteria>
  </DataArea>

</ShowBatchInformation>
```

Request: ProcessDCSEvent

The request for sending a DCS event to an MES is an XML document of the **ProcessDCSEvent** type. The DCS event is located in the **DataArea** container (page 26):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
- **DataArea/ControlRecipe** contains the list of requested batch values:
  - **BatchID** contains the target batch of the order.
  - **MaterialDefinitionID** contains the name of the material of the target batch of the order.
  - **MessageEvent/Identifier** contains the identifier of the event.
  - **MessageEvent/AsynchronousUsage** indicates whether the message is sent asynchronously (**true**) or not (**false**).

A DCS event is sent for synchronization purposes. In the synchronous use case, the receiver must be active when the request is sent. In the asynchronous use case, in which no reply is sent, it is not necessary that the receiver is running when the request is sent. So it makes sense to define a longer time to live (page 15) of the message than in the synchronous use case. The time to live of the message must be the period in which the receiver becomes active. Example: It is assumed that the receiver becomes active within 24 hours. The time to live must be set to 24 * 60 * 60 * 1000.

Example of a 'process DCS event' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ProcessDCSEvent
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>DeltaV B1</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>
  <DataArea>
    <BatchID>ReleasedBatch</BatchID>
    <MaterialDefinitionID>Shiny pills</MaterialDefinitionID>
    <MessageEvent>
      <Identifier>AnyEventIdentifier</Identifier>
      <AsynchronousUsage>false</AsynchronousUsage>
    </MessageEvent>
  </DataArea>
</ProcessDCSEvent>
```

Reply: AcknowledgeDCSEvent

The reply to the synchronous **ProcessDCSEvent** request is an XML document of the **AcknowledgeDCSEvent** type. The results are located in the **DataArea** container (page 26):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.

Example of an Accepted reply to a 'process DCS event' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeDCSEvent
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Accepted"/>
  </DataArea>

</ AcknowledgeDCSEvent>
```

Example of a Rejected reply to a 'process DCS event' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeDCSEvent
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected”>
      An error has been occurred.
    </ResponseCriteria>
  </DataArea>

</ AcknowledgeDCSEvent>
```

Request: ProcessDCSParameters

The request for sending DCS parameters to a DCS is an XML document of the **ProcessDCSParameters** type. The DCS parameters are located in the **DataArea** container (page 26):

- **DataArea** contains the pre-defined set of parameters in the correspondingly named elements.
  - **BatchID** contains the batch of the order.
  - **RecipeID** contains the identifier of the recipe.
  - **MaterialDefinitionID** contains the name of the material of the batch of the order.
  - **EquipmentID** contains the identifier of the unit.

- **EquipmentList** contains the list of equipment and control module IDs.
- **DataTarget** contains the data target to define how the message is processed in the Manufacturing Service Bus.

- **DataArea/ControlRecipe** contains the list of DCS parameter values:
  - **Parameter** contains a DCS parameter.
    - **ID** contains the identifier of the parameter.
    - **Categorization** contains the category of the parameter.
    - **Value<data type>** contains the type-specific value of the parameter.

Example of a 'process DCS parameters' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ProcessDCSParameters
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>JavaDCSMock</Receiver>
  </ApplicationArea>
  <DataArea>
    <CreationDate>2018-12-05T11:07:21.502+01:00</CreationDate>
    <BatchID>BatchGood</BatchID>
    <RecipeID>AnyRecipe</RecipeID>
    <MaterialDefinitionID>AnyMaterial</MaterialDefinitionID>
    <EquipmentID>AnyEquipment</EquipmentID>
    <EquipmentList>
      <ActualEquipmentID>controlModule01</ActualEquipmentID>
      <ActualEquipmentID>controlModule10</ActualEquipmentID>
    </EquipmentList>
     DataTarget>AnyDataTarget</DataTarget>
    <ControlRecipe>
      <Parameters>
        <Parameter>
          <ID>STATUS</ID>
          <ValueString>OK</ValueString>
          <Categorization>P1</Categorization>
        </Parameter>
        <Parameter>
          <ID>START_TIME</ID>
          <ValueDatetime>2018-12-05T11:07:21.515+01:00</ValueDatetime>
          <Categorization>P2</Categorization>
        </Parameter>
        <Parameter>
          <ID>A_DURATION</ID>
          <ValueDuration>P5DT12H35M30S</ValueDuration>
          <Categorization>P3</Categorization>
        </Parameter>
        <Parameter>
          <ID>LEVEL</ID>
          <ValueInteger>42</ValueInteger>
          <Categorization/>
```

```
        </Parameter>
        <Parameter>
          <ID>FLAG</ID>
          <ValueBoolean>true</ValueBoolean>
          <Categorization/>
        </Parameter>
        <Parameter>
          <ID>CHARGE_ACTUAL</ID>
          <ValueNumeric>3.14</ValueNumeric>
          <Categorization>P1</Categorization>
        </Parameter>
      </Parameters>
    </ControlRecipe>
   /DataArea>
</ProcessDCSParameters>
```

Reply: AcknowledgeDCSParameters

The reply to the **ProcessDCSParameters** request is an XML document of the **AcknowledgeDCSParameters** type. The results are located in the **DataArea** container (page 26):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.

Example of an Accepted reply to a 'Process DCS parameters' request

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeDCSParameters
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>JavaDCSMock</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>
  <DataArea>
    <ResponseCriteria actionCode="Accepted"/>
  </DataArea>
  <OtherInformations/>
</AcknowledgeDCSParameters>
```

Request: GetDCSParameters

The request for getting DCS parameters from a DCS is an XML document of the **GetDCSParameters** type. The DCS parameters are located in the **DataArea** container (page 26):

- **DataArea** contains some common data and the pre-defined set of parameters in the correspondingly named elements.
  - **BatchID** contains the batch of the order.
  - **RecipeID** contains the identifier of the recipe.
  - **MaterialDefinitionID** contains the name of the material of the batch of the order.
  - **EquipmentID** contains the identifier of the unit.
  - **EquipmentList** contains the list of equipment and control module IDs.
  - **DataSource** contains the data source to define how the message is processed in the Manufacturing Service Bus.
- **DataArea/ControlRecipe** contains the list of DCS parameters to be get:
  - **RecipeElement** contains a DCS parameter.
    - **ParameterPath** contains the path of the parameter.
    - **ParameterID** contains the identifier of the parameter.
    - **ParameterSubType** contains the subtype of the parameter.
    - **DataType** contains the data type of the value of the parameter.

Example of a 'get DCS parameters' request

```
<?xml version="1.0" encoding="UTF-8"?>

<GetDCSParameters
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>JavaDCSMock</Receiver>
  </ApplicationArea>
  <DataArea>
    <CreationDate>2018-12-05T12:42:19.977+01:00</CreationDate>
    <BatchID>BatchGood</BatchID>
    <RecipeID>AnyRecipe</RecipeID>
    <MaterialDefinitionID>AnyMaterial</MaterialDefinitionID>
    <EquipmentID>AnyEquipment</EquipmentID>
    <EquipmentList>
      <ActualEquipmentID>controlModule01</ActualEquipmentID>
      <ActualEquipmentID>controlModule10</ActualEquipmentID>
    </EquipmentList>
```

```
    <DataSource>AnyDataSource</DataSource>
    <ControlRecipe>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>STATUS</ParameterID>
        <ParameterSubType>Report</ParameterSubType>
        <DataType>string</DataType>
      </RecipeElement>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>CHARGE_ACTUAL</ParameterID>
        <ParameterSubType>Report</ParameterSubType>
        <DataType>decimal</DataType>
      </RecipeElement>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>LEVEL</ParameterID>
        <ParameterSubType>Prompt</ParameterSubType>
        <DataType>integer</DataType>
      </RecipeElement>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>FLAG</ParameterID>
        <ParameterSubType>Recipe</ParameterSubType>
        <DataType>boolean</DataType>
      </RecipeElement>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>START_TIME</ParameterID>
        <ParameterSubType>StatusChange</ParameterSubType>
        <DataType>dateTime</DataType>
      </RecipeElement>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>A_DURATION</ParameterID>
        <ParameterSubType xsi:nil="true"/>
        <DataType>duration</DataType>
      </RecipeElement>
    </ControlRecipe>
  </DataArea>
</GetDCSParameters>
```

Reply: ShowDCSParameters

The reply to the **GetDCSParameters** request is an XML document of the **ShowDCSParameters** type. The results are located in the **DataArea** container (page 26):

- **DataArea/ResponseCriteria/actionCode** contains **Accepted** or **Rejected**.
- **DataArea/ResponseCriteria** contains an error message if **actionCode** contains **Rejected**.
- **DataArea/ControlRecipe** contains the list of returned DCS parameter values:
  - **RecipeElement/ParameterPath** contains the path of the parameter.
  - **RecipeElement/ParameterID** contains the name of the batch value.

- **RecipeElement/ParameterSubType** contains the subtype of the parameter.
- **RecipeElement/Value<data type>** contains the returned DCS parameter value of the requested data type.

- **DataArea/Table** contains a table with header row, up to 50 data rows, and up to 5 columns. Header and data are defined by the DCS.

Example of an Accepted reply to a 'get DCS parameters' request

```
<?xml version="1.0" encoding="UTF-8"?>

<ShowDCSParameters
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>JavaDCSMock</Sender>
    <Receiver>MES</Receiver>
  </ApplicationArea>
  <DataArea>
    <ResponseCriteria actionCode="Accepted"/>
    <ControlRecipe>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>CHARGE_ACTUAL</ParameterID>
        <ParameterSubType>Report</ParameterSubType>
        <ValueNumeric>3.14</ValueNumeric>
      </RecipeElement>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>FLAG</ParameterID>
        <ParameterSubType>Recipe</ParameterSubType>
        <ValueBoolean>false</ValueBoolean>
      </RecipeElement>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>START_TIME</ParameterID>
        <ParameterSubType>StatusChange</ParameterSubType>
        <ValueDatetime>2018-12-05T12:42:20.251+01:00</ValueDatetime>
      </RecipeElement>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>STATUS</ParameterID>
        <ParameterSubType>Report</ParameterSubType>
        <ValueString>aString</ValueString>
      </RecipeElement>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>A_DURATION</ParameterID>
        <ParameterSubType xsi:nil="true"/>
        <ValueDuration>P5DT12H35M30S</ValueDuration>
      </RecipeElement>
      <RecipeElement>
        <ParameterPath>UP_PV123_/OP_PV123/P_PV_NOAH</ParameterPath>
        <ParameterID>LEVEL</ParameterID>
        <ParameterSubType>Prompt</ParameterSubType>
        <ValueInteger>3</ValueInteger>
      </RecipeElement>
```

```
    </ControlRecipe>
    <Table>
      <TableLayout>3 Columns</TableLayout>
      <TableHeader>
        <Header1>Header1</Header1>
        <Header2>Header2</Header2>
        <Header3>Header3</Header3>
      </TableHeader>
      <TableRow>
        <sequenceNr>1</sequenceNr>
        <Column1>Row1 Column1</Column1>
        <Column2>Row1 Column2</Column2>
        <Column3>Row1 Column3</Column3>
      </TableRow>
      <TableRow>
        <sequenceNr>2</sequenceNr>
        <Column1>Row2 Column1</Column1>
        <Column2>Row2 Column2</Column2>
        <Column3>Row2 Column3</Column3>
      </TableRow>
    </Table>
  </DataArea>
  <OtherInformations/>
</ShowDCSParameters>
```

Specifics Related to the Process DCS Event Touchpoint

The JMS message of the *Process DCS Event* touchpoint must additionally contain a message property of the *String* type:

- Name of the Property: MessageContextPropertyKey
- Value:
  The value of the property is a concatenation of the attribute values sent in the xml (page 42) separated by a slash. The common pattern of the value is:
  <BatchID>/<MaterialDefinitionID>/<MessageEvent identifier>
  Example: ReleasedBatch/Shiny pills/AnyEventIdentifier

TIP

Unless the property is set in the message, no listener will receive the message.

A listener for the *Process DCS Event* touchpoint runs in a phase building block of FT PharmaSuite that is configured for a specific event identifier. There can be multiple listeners for the *Process DCS Event* touchpoint. This depends on the recipe and the number of orders running in the MES. Each listener listens only for a specific batch, material, and event identifier combination. The property is used as JMS selector.

In case the DCS erroneously sends messages for a batch, material, and event identifier and there is no corresponding listener, the messages will remain in the queue until they

have expired. The expiry date is defined with the *MessageTimeToLive* configuration parameter (page 15).

In case the DCS sends a message and no phase building block is running that waits for the message, the message will also remain in the queue. However, if the phase becomes active at a later point in time during the period until the message expires, the phase receives the message. In the synchronous use case this message receipt is unexpected at that point in time and, in addition, the DCS might have handled the message as an error since no reply has been received within the expected time frame.

However, this is the expected and intended behavior of the asynchronous use case. The period of time the message shall live thus needs to be longer than in the synchronous use case. In the asynchronous use case, the time to live (page 15) of the message must be set to the maximum period of time the listener remains active after the event has been sent.

TIP

To avoid unexpected behavior and prevent the JMS queue from becoming very large or even full, we highly recommend to specify the expiration of the DCS Event JMS message.

After a restart of the ActiveMQ broker all messages that were available in the JMS queue before the restart are lost. In case event messages were lost, the corresponding listener will never receive the messages and must be aborted.

Example of an MSB Implementation Based on EIHub

This example describes how to structure an integration flow in the Manufacturing Service Bus. The example is based on Enterprise Integration Hub (EIHub). However, the overall approach is similar for other MSBs/ESBs.

The figure below illustrates the flow for integrating the PlantPAx® DCS, which provides a RESTful web service interface for batch creation. The flow only handles the **Create DCS Batch** integration touchpoint. The other touchpoints are not included. For including them, a similar structure may be needed.

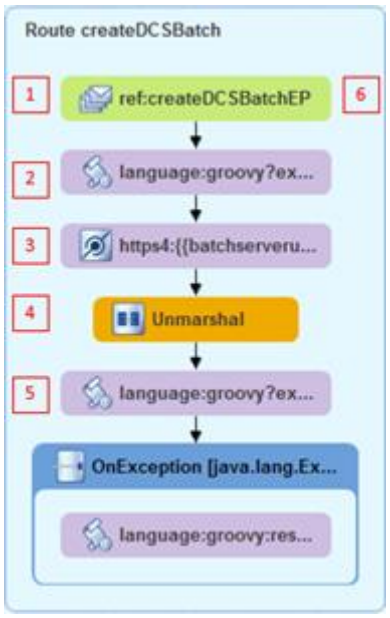

Figure 7: Example of an MSB flow

The flow consists of the following steps:

1. The JMS endpoint accepts the request message from the DCS Adapter.
   The message contains the XML structure as described in "General Structure of the XML Requests and Replies" (page 26).
2. The data transformation script converts this XML document into a JSON document that complies with the JSON data structure for the PlantPAx RESTful service.
3. The RESTful web service offered by the PlantPAx DCS is called.
4. The RESTful web service returns the result of the action in the DCS.
5. The vendor-specific result structure is converted back into the XML schema expected by the DCS Adapter for the reply, by means of an appropriate script.
6. The JMS reply is returned to the DCS Adapter

The figure below illustrates an alternative approach with a mock flow. For simulation purposes, you can also set up a mock flow in the MSB without a real connection to a DCS. In this case, you only log the message received from the DCS Adapter and perform a simple conversion into the reply XML structure.

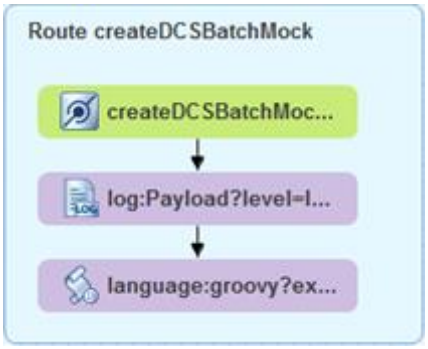

Figure 8: Example of an MSB mock flow

Testing the DCS Integration without an MES Client

To test the DCS integration without an MES client, you can use the **TestDCSAdapter** tool provided for this purpose (see section "Testing the MES Client without a DCS" (page 18)).

The **TestDCSAdapter** tool contains tabs for each integration touchpoint. The *Process Consumed Material* and *Process Produced Material* tabs are combined in the *Material-related* tab. In each tab, you can provide the data needed to build a request. The touchpoint-specific **Process** button will trigger the DCS Adapter to send the corresponding request to the given receiver. Thus, you trigger a request message to be sent to your DCS.

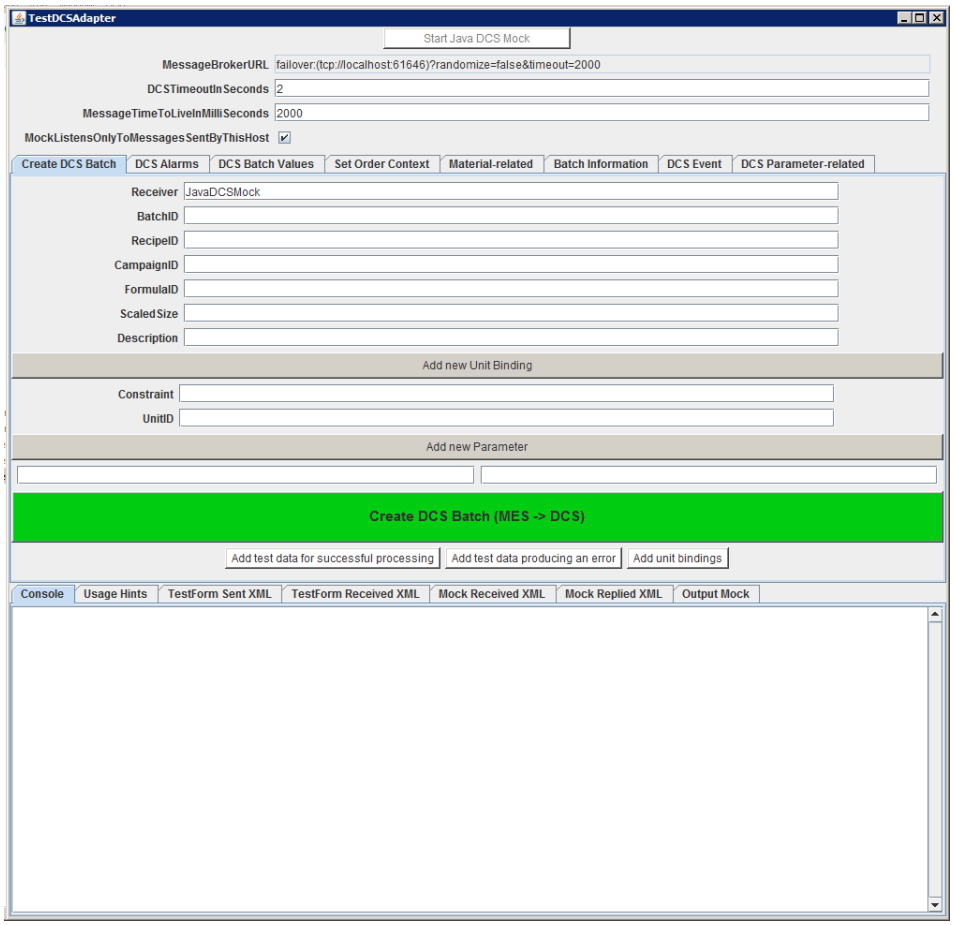

Figure 9: Create DCS batch tab of the TestDCSAdapter tool

The upper part of the form contains the configuration of the DCS Adapter. Here you can configure the message broker URL.
In the **Receiver** box, type the name of your DCS. In the additional boxes, you can enter further test data.
When you have completed the data for your request, click the touchpoint-specific **Process** button to trigger the DCS Adapter to send the corresponding request.

Extending the Standard

If there is a need to transfer further data that is not contained in the standard, it is possible to add a list of key/value pairs to the XML document for both directions. The *IExtendedDCSService* interface provides methods for each integration touchpoint. Each method has an additional parameter for a map of keys and values. Furthermore, the return value of these methods returns a map that contains the key/value pairs included in the reply message.

In the XML document, the additional data is located in the **OtherInformations** container (page 26).

Example of a 'create process XML document with OtherInformations'

```
<?xml version="1.0" encoding="UTF-8"?>
<ProcessCreateDCSBatch
        xmlns="http://www.rockwell.com/mes/dcs/ifc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>Building_API_220</Receiver>
  </ApplicationArea>

  <DataArea>
    <BatchID>BS101338_27Oct15_2111_PV51400</BatchID>
    <MasterRecipeID>UP_BUFP_BUFFER_MAKEUP_MP</MasterRecipeID>
      <ControlRecipe>
        <Parameters> </Parameters>
        <EquipmentRequirements> </EquipmentRequirements>
      </ControlRecipe>
      <CampaignID>DeltaV_V11_CAMP</CampaignID>
      <ScaledSize>100.00</ScaledSize>
      <FormulaID>SFPHC_PV515400_MAKEUP</FormulaID>
    <Description>This is a long description</Description>
  </DataArea>

  <OtherInformations>
    <OtherInformation>
      <ID>AdditionalData01</ID>
      <ValueString>AValue</ValueString>
    </OtherInformation>
      <OtherInformation>
      <ID>AdditionalData02</ID>
      <ValueDatetime>2016-01-14T14:54:43</ValueDatetime>
    </OtherInformation>

  </OtherInformations>
</ProcessCreateDCSBatch>
```

Example of a reply message with 'OtherInformations'

```
<?xml version="1.0" encoding="UTF-8"?>

<AcknowledgeCreateDCSBatch
       xmlns="http://www.rockwell.com/mes/dcs/ifc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemeVersionID="2.4">

  <ApplicationArea>
    <Sender>MES</Sender>
    <Receiver>DeltaV B1</Receiver>
  </ApplicationArea>

  <DataArea>
    <ResponseCriteria actionCode="Rejected">
      Error during batch creation. Batch ID already exists.
    </ResponseCriteria>
  </DataArea>

  <OtherInformations>
    <OtherInformation>
      <ID>Custom field: DeltaV Internal Id</ID>
      <ValueString>DV_323433232</ValueString>
    </OtherInformation>
  </OtherInformations>

</AcknowledgeCreateDCSBatch>
```

Extending Standard Implementation of Listeners in an MES Client

To answer and handle requests sent by a DCS to an MES Client, an MES must implement the correspondent listeners. For details, see "Implementing Listeners" (page 13).

The DCS Adapter provides the possibility to transfer further data in a request. To use the additional data in the implementation of the MES, the MES must be customized. For details, please refer to the documentation of the respective MES client. For FT PharmaSuite, see "Material Handling for DCS" in Volume 3 of the "Technical Guide Configuration and Extension" [A3] (page 59).

The listener for *Process DCS Event* requests is located in phase building blocks of FT PharmaSuite. In case the listener has to be adapted, the corresponding phase building block must be adapted.

Chapter 6

Upgrade-related Topics

The following list of topics refers to changes that are related to changed or deprecated functionality of the DCS Adapter.

IDCSAdapterConfiguration

Updated for DCS Adapter 2.5.

- **Jira ID:** CVBLSFTPS-15805
- **Before:** The interface contains two methods regarding timeouts: getReplyTimeoutInSeconds, used as message timeout and reply timeout and getMessageTimeToLive, used as message TTL.
- **After:** The getMessageTimeToLive method is used as message TTL and reply timeout. In the standard implementation of the interface, the default value is 300000 ms (5 minutes). The getReplyTimeoutInSeconds is deprecated. There is a new method called getMessageTimeoutInSeconds, which is used as message timeout.
- **Reason:** The method names were misleading. The message timeout is usually a small value (default in PS is 2 sec) which is insufficient for a reply timeout.
- **Migration:** use getMessageTimeoutInSeconds() for the message timeout and getMessageTimeToLive() for the message TTL and reply timeout (please note that the value of getMessageTimeToLive() is in milliseconds).

Reference Documents

The following documents are available from the Rockwell Automation Download Site.

No.	Document Title	Part Number
A1	FT PharmaSuite Technical Guide Installation	PSES-IN011A-EN-E
A2	FactoryTalk ProductionCentre Plant Operations Release 10.4 Server Installation Guide - JBoss Advanced	PCJBAD IN104A EN E
A3	FT PharmaSuite Technical Guide Configuration & Extension - Volume 3	PSCEV3-GR011A-EN-E

TIP

To access the Rockwell Automation Download Site, you need to acquire a user account from Rockwell Automation Sales or Support.

Appendix A

Revision History

The following tables describe the history of this document.

Changes related to the document:

Object	Description	Document
---	---	---

Changes related to "Introduction" (page 1):

Object	Description	Document
---	---	---

Changes related to "Architecture of an MES Integrated with the DCS Adapter" (page 5):

Object	Description	Document
---	---	---

Changes related to "Integration into an MES" (page 9):

Object	Description	Document
Configuring the DCS Adapter (page 15)	Updated MessageBrokerURL example with "ssl://hostname:61647".	1.0
Simulating a DCS with a Java Mock (page 16)	Updated MessageBrokerURL example with "ssl://hostname:61647".	1.0

Changes related to "Integration of a DCS" (page 23):

Object	Description	Document
---	---	---

Changes related to "Extending the Standard" (page 55):

Object	Description	Document
---	---	---

Changes related to "Upgrade related Topics (page 57)":

Object	Description	Document
---	---	---

Index

S

T

U

X